

Re: [Lit.] Buffer overruns

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2005-01/0203.html>

From: Anne & Lynn Wheeler (lynn_at_garlic.com)

Date: 01/03/05

Date: Sun, 02 Jan 2005 17:16:14 -0700

"John E. Hadstate" <jh113355@hotmail.com> writes:

- > *If you recall, there were two bits for each memory region:*
- > *read-enable and write enable. No-read, No-write meant*
- > *no-access. Write-no-read meant I-fetch-only access if the*
- > *processor was in non-privileged state. It meant read-write*
- > *if the processor was in privileged state.*
- >
- > *Your Big Blue credentials notwithstanding. I spent many*
- > *nights reading the system architecture documents for S/360,*
- > *some of which were intended for distribution only to Field*
- > *Service engineers, during the summer of 1970. The*
- > *architecture did have support for no-access, read-only,*
- > *read-write, and execute only. Whether the hardware for a*
- > *particular model of S/360 supported it depended on other*
- > *(sometimes unreasonable) things, such as the presence of*
- > *floating point support or a memory expansion option.*
- >
- > *You are right about the lack of VM support in S/360. O/S*
- > *360 MFT could have multiple partitions, but the linker did*
- > *all the relocation work up front. However, I believe the*
- > *S/360 Model 91 did have virtual memory support. To be fair,*
- > *I believe NASA was the only customer for it and it had*
- > *problems and a price tag that made it uninteresting to the*
- > *commercial world. Clemson University had a S/370-155, then*
- > *a 165 in 1970 that had virtual memory. IBM sales tried to*
- > *sell me a S/370-110 in 1972 that had virtual memory*
- > *hardware. They were still peddling DOS for it which, by*
- > *that time, had virtual memory support grafted on.*
- >
- > *Self-modifying application code became an interesting*
- > *problem in the 360 Model 91 (was there also a 61) because of*
- > *pipelined prefetch. IBM's official (and documented) answer*
- > *was, "Don't do that." The 360/370 instruction set included*
- > *an "Execute" instruction for modifying fields (mostly length*
- > *fields) in arbitrary instructions outside the instruction*
- > *stream without actually changing the contents of memory*
- > *where the modified instruction was fetched from. (This*

- > *presented some interesting problems if the target*
- > *instruction crossed a page boundary and one of the pages was*
- > *missing.) Thus, while the code could be self-modifying in a*
- > *sense, it didn't have to be as bad as it sounds. Of course,*
- > *that's not to say that some people didn't get away with it,*
- > *especially on small 360's running DOS or TOS where memory*
- > *management was pretty loose.*

there might be lots of stuff that could be model specific ... but because they weren't in general use ... didn't gain support as part of the standard programming paradigm.

the original virtual memory on 360 was done on a specially modified 360/40 ... but as far as I know, only one such machine was built. This was built for the virtual memory/machine project at the science center, 4th floor, 545 tech sq

<http://www.garlic.com/~lynn/subtopic.html#545tech>

The standard virtual memory offering for 360 was the 360/67, which also supported both 24-bit and 32-bit virtual addressing modes. It basically was a 360/65 with virtual memory hardware support added (and in the smp case, other types of stuff). The official corporate virtual memory operating system for the 360/67 was tss/360. Because of complexity, performance, and implementation issues, a lot of places took their 360/67 and used them for other stuff. Some number of places just ran their 360/67 in straight 360/65 mode with batch operating system. UofM wrote the Michigan Terminal System (MTS) which some number of locations ran on 360/67. The majority of the 360/67s eventually were running the virtual machine/memory system developed at the science center (the center converted the system from the custom modified 360/40 to 360/67 when the machine became available) ... or some commercial derivative of the same ... aka IDC, NCSS, etc for commercial time-sharing service bureaus, misc. past postings

<http://www.garlic.com/~lynn/subtopic.html#timeshare>

There are some similarities about the unic/C split off from Multics and the virtual memory/machine system done by the science center vis-a-vis the official corporate strategic operating system TSS/360.

The folklore is that the original announced products were 360/60, 360/62 and 360/70 ... all machines with real storage that cycled at one microsecond. I don't know if any such machines actually shipped. These machines were quickly replaced with the announcement of the 360/65, 360/67, and 360/75 ... which were the machines upgraded with real storage that cycled at 750ns.

I never heard of 61. There was 91, 92, 95, 360/195 and 370/195. There was also a 360/44 ... sort of subset of 360/40 instructions/hardware with enhanced floating point hardware performance.

I had some involvement with a project that looked at doing a 370/195 that emulated two processor machine ... doing red/black tagging of registers and instructions in the pipeline. The problem with these pipeline machines was that a (almost all) branch instruction would drain the pipeline (no branch prediction &/or speculative execution). Except for very specialized codes, 370/195 pipeline rarely got more than half-full before encountering a branch instruction (resulting it running at something like half the peak rated mip rate). Simulating two-processor machine with dual i-streams (something like the modern day hyperthreading support), could have two i-streams, each keeping the 370/195 i-stream half-full ... which might amount to a full pipeline.

The execute instruction took another instruction as an argument and used a parameter from the execute instruction to modify the 2nd byte in the target instruction. This was the length field in the character and decimal instructions and the "immediate" field, in the immediate instructions. The 360/67 required a minimum of 8 hardware relocate look-aside registers ... since the worst case minimum instruction execution could require 8 different addresses:

- 2; execute instruction crosses a page boundary,
 - 2; target instruction crosses a page boundary,
 - 2; target is "ss" instruction (aka storage to storage) with two storage address operands
 - 2; target is "ss" nstruction with both storage operands crossing a page boundary (i.e. precalculates both start and end of storage operands)
- =
8

The maximum possible length of ss instruction operands was 256byte, instruction decode only needed to preresolve the start and end address (which wouldn't cross more than one page boundary using either 2k pages or 4k pages). on 360 and most 370 instructions, instruction decode would resolve both starting and ending operatnd address ... and test for access permission. If there was wasn't access permission for all storage references (a single in all but ss, and two in ss-instruction case) for both the starting and ending addresses of the resepective storage location, there would be a hardware interrupt before start of instruction execution.

It might be observed that this may have helped promote software paradigms that kept track of all lengths ... since there were no standard machine instructions that supported implied lengths.

Images of executable code on disk included control information about address location dependencies in the code. The link/loader would modify the executable code image after it was brought into storage to adjust address location dependencies. This loader/link function was independent of whether you ran PCP, MFT, or MVT. Even in the purely PCP environment running only a single application at a time ... an

application program executable image would be loaded (and the address dependencies adjusted) ... and then any library program images tended to get loaded after the application program code. Since the application program code could be of arbitrary size ... the load location of library code could be somewhat arbitrary. Quite a few of the address location dependencies were address constants of subroutine entry points that tended to be intermixed with code. A specific PCP release would tend to have a constant starting address for loading application code ... but things like library code would be loaded and somewhat arbitrary locations (dependent on the size of the application being run). MFT (& MVT) could have multiple different addresses for starting the load of application code (but the link/loader already had to default for arbitrary address adjustments because of the generalize issue of loading library code). This is characteristic is somewhat orthogonal to having a pointer passing paradigm vis-a-vis possibly value passing paradigm, whether standard programming model allowed for execute only, read only, no execute, etc. however, for lots of postings on the address dependency problem with shared, r/o code <http://www.garlic.com/~lynn/subtopic.html#adcon>

The first engineering 370 with virtual memory support was the 370/145. The standard front panel of all 370/145s had a light labeled XLATE (long before virtual memory was announced for the 370). There was lots of customer speculation that "XLATE" stood for translate-mode (aka address relocation support virtual address spaces).

This machine was used to port the virtual machine/memory operating system done by the science center from 360/67 to 370 (there was some number of differences between the 360/67 and 370 virtual memory architecture, including the 370 architecture only supported 24-bit address mode and didn't support 32-bit address mode). The initial prototype of the standard corporate batch operating system involved taking a straight forward MVT operating system ... and crafting a small virtual address space layer quite a bit of it from bits & pieces from the science center operating system. For the most part the existing MVT operating system code ... continued to operate as if it was running on a real storage machine that happened to have 16mbytes of memory.

Actually, the science center had a joint project with the endicott engineers responsible for the 370/145 virtual memory hardware. The standard 360/370 hardware reference manual is called the principle of operations ... which has detailed description and operation of instructions had infrastructure for 360/370. It has traditional been (since the late '60s) a machine readable file that is part of something called the architecture manual "red-book" (because it was traditional distributed in a red 3ring binder). The file could be printed/formated under command control as the full architecture manual ... or as the publicly available principle of operation subset (with conditional formatting controls). The full architecture manual tended to have lots of engineering notes, justifications for the instruction

or feature, and various trade-off discussions ... like why some things might be feasible with a specific hardware technology ... but not possible as part of the official 360/370 architecture (because it possibly wasn't practical across all hardware technology used in the various different processor models). In any case, special operating system software was crafted to emulate the full virtual memory 370 architecture ... running on a 360/67. The result was 370 virtual machines that could be used for testing (even tho the 360/67 and 370 virtual memory architecture had numerous differences). This project had virtual 370 machines up and running production a year before the first engineering virtual memory 370 machine was operational.

principle of operations is a generally available customer manual (current versions for 390 and Z series have been available online)

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/DZ9AR004/CCONTENTS?SHELF=EZ2HW125&

it is unlikely the architecture red-book version was available in the field.

for specific models, you could also get the functional specification manual. Some number of the 360 functional characteristics manuals have been scanned and made available online:

<http://bitsavers.org/pdf/ibm/360/>

including 40, 44, 65, 67, 91, 195, etc, the 360 functional characteristic manuals tended to include detailed manual specific things like instruction timings. numerous other early hardware and software manuals have been scanned and are online

For each specific machine, FEs would also have detailed wiring and microcode implementation manuals ... typically located in the machine room near the machine. Specific machine models might have implementation for stuff not prescribed by 360/370 architecture

at the above site, there are (at least) "-0", "-6", and "-7" of the principle of operations manual. Page 17 has short description of "protection features" ...which might be available on a 360 machine, one is store protection feature and the other is fetch protection feature. I believe for OS/MFT (OS/MVT, and later) the only required feature was store protect (I don't believe the operating system and/or any other software required fetch protection feature to be available for operation).

Prior to MFT (and then MVT) requiring store protect feature, it was possible for applications to stomp on low-storage and do things like enter privilege/supervisor state ... possibly an exploit, but also a mechanism used by some number of regular application ... like hasp ... some number of past hasp postings

<http://www.garlic.com/~lynn/subtopic.html#hasp>

Note, it doesn't mention execute-only as any standard 360 feature. However 360 architecture wouldn't preclude the 360/40 engineers from

implementing such a feature ... it just wasn't part of 360 (and wouldn't likely be used by any standard programming paradigm). 360/40s typically had other stuff that weren't part of 360 ... like microcode for hardware emulation of 1401/1410/etc ... as an alternative to the microcode engine emulating 360 instructions ... typically controlled by some switch on the front panel ... a picture of real 1401:

http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_2423PH2044.html

picture of 360/40 (there should be some sort of switch on the front panel ... probably lower center cluster that could select between 360 instruction microcode or 1401 instruction microcode operation of the machine):

http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_2423PH2040.html

a picture of similar 360/44 used for floating point/scientific workloads:

http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_2423PH2044.html

the following reference has some amount material about the evolution of CTSS to multics ... project mac choosing the GE machine instead of the ibm 360/67 hardware, and various and sundry tidbits about tss/360 ... but mostly about the virtual memory/machine operating system developed by the science center (initially on a custom modified 360/40 with virtual memory and later converted to standard 360/67 product line machine)

<http://pucc.princeton.edu/~melinda/>

The "low-end" 370 was the 115/125 models developed by Boebllingon. Except for a few exceptions, the 360 and 370 machines were microcoded implementations. On the low & mid-range 370s ... the native processor engine typically executed an avg. of 10 instructions for every 370 instruction. The 115/125 hardware implementation used a 9 position, shared memory buses ... which could have microprocessor engines installed. One microprocessor would have the support for 370 instruction set loaded and the other (typically) 3-8 microprocessor engines would have other microcode installed to handle various control and I/O functions. In the 115 all the microprocessors were identical (except for the microcode load). The 125 was identical to a 115 except the microprocessor engine that had the 370 microcode load was approx. 25% faster than the other engines. The 115 was rated at about 80KIPS 370 (requiring a native processor engine of about 800KIPS) and the 125 was rated at about 100KIPS 370 (requiring a native processor engine of about 1MIPS).

a site listing most of the 360 and 370 models with original announce data as well as first customer ship (including 360/60, 360/62, 360/70 announce dates):

<http://www.isham-research.com/chrono.html>

per above reference, 370/165 (w/o virtual memory) was announced June, 1970 and first customer ship of a machine was spring of 1971.

and for some clemson related topic drift:

<http://www.cs.clemson.edu/~mark/fs.html>

misc. other clemson references that somebody might find of interest:

http://www.cs.clemson.edu/~mark/acs_technical.html

http://www.cs.clemson.edu/~mark/acs_timeline.html

also per above, misc 370 announce & ship dates: 370/125 ANN 10/72, first ship, spring 73; 370/115 ANN 3/73, first ship spring 74.

370 virtual storage was announced 8/72 and immediately available on 370/135 and 370/145 with new microcode load .. but required purchase of field installable hardware to add virtual memory to 370/155 and 370/165. Operating systems were dos/vs (virtual storage version of dos), vs1 (somewhat virtual storage version of mft), and vs2 (virtual storage version of mvt).

There was some argument with the 360/165 engineers about implementing the full 370 architecture (as per the "red-book"); they claimed it would had six month delay to the schedule to add support for selective invalidate as well as segment r/o protect. They won since it was more important to get virtual storage announced and in the field than to make sure that all features of the architecture were included (the excluded features then had to be removed from the 135 & 145 implementations that already existed).

minor summary

8/70 370/165 announced
2q/71 370/165 FCS .. first customer ship
8/72 virtual memory/storage announced
10/72 370/125 announced
3/73 370/115 announced
2q/73 370/125 FCS
2q/74 370/115 RCS

short description of 360 & 370 machine models:

<http://www.beagle-ears.com/lars/engineer/comphist/model360.htm>

30 year history of MTS:

<http://www.clock.org/~jss/work/mts/30years.html>

random past references to 370/195 dual i-stream project:

<http://www.garlic.com/~lynn/94.html#38> IBM 370/195

<http://www.garlic.com/~lynn/95.html#3> What is an IBM 137/148 ???

<http://www.garlic.com/~lynn/99.html#73> The Chronology

<http://www.garlic.com/~lynn/99.html#97> Power4 = 2 cpu's on die?

<http://www.garlic.com/~lynn/2000g.html#15> 360/370 instruction cycle time

<http://www.garlic.com/~lynn/2001b.html#38> Why SMP at all anymore?

<http://www.garlic.com/~lynn/2001j.html#27> Pentium 4 SMT "Hyperthreading"

<http://www.garlic.com/~lynn/2001n.html#63> Hyper-Threading Technology – Intel information.

<http://www.garlic.com/~lynn/2002g.html#70> Pipelining in the past

sci.crypt: Re: [Lit.] Buffer overruns

<http://www.garlic.com/~lynn/2002g.html#76> Pipelining in the past
<http://www.garlic.com/~lynn/2003f.html#33> PDP10 and RISC
<http://www.garlic.com/~lynn/2003l.html#48> IBM Manuals from the 1940's and 1950's
<http://www.garlic.com/~lynn/2003m.html#60> S/360 undocumented instructions?
<http://www.garlic.com/~lynn/2003p.html#3> Hyperthreading vs. SMP
<http://www.garlic.com/~lynn/2004e.html#1> A POX on you, Dennis Ritchie!!!
<http://www.garlic.com/~lynn/2004.html#27> dual processors: not just for breakfast anymore?
<http://www.garlic.com/~lynn/2004o.html#18> Integer types for 128-bit addressing

random past references to UofM MTS (virtual memory operating system
built for the 360/67 and later ported to 370 virtual memory)

<http://www.garlic.com/~lynn/2000b.html#61> VM (not VMS or Virtual Machine, the IBM sort)
<http://www.garlic.com/~lynn/2000c.html#44> WHAT IS A MAINFRAME???
<http://www.garlic.com/~lynn/2000f.html#52> TSS ancient history, was X86 ultimate CISC? designs)
<http://www.garlic.com/~lynn/2000g.html#0> TSS ancient history, was X86 ultimate CISC? designs)
<http://www.garlic.com/~lynn/2000.html#91> Ux's good points.
<http://www.garlic.com/~lynn/2001m.html#55> TSS/360
<http://www.garlic.com/~lynn/2001n.html#45> Valid reference on lunar mission data being unreadable?
<http://www.garlic.com/~lynn/2002i.html#63> Hercules and System/390 – do we need it?
<http://www.garlic.com/~lynn/2002n.html#54> SHARE MVT Project anniversary
<http://www.garlic.com/~lynn/2002n.html#64> PLX
<http://www.garlic.com/~lynn/2003b.html#0> Disk drives as commodities. Was Re: Yamhill
<http://www.garlic.com/~lynn/2003b.html#10> Disk drives as commodities. Was Re: Yamhill
<http://www.garlic.com/~lynn/2003f.html#41> SLAC 370 Pascal compiler found
<http://www.garlic.com/~lynn/2003j.html#54> June 23, 1969: IBM "unbundles" software
<http://www.garlic.com/~lynn/2003k.html#5> What is timesharing, anyway?
<http://www.garlic.com/~lynn/2003l.html#30> Secure OS Thoughts
<http://www.garlic.com/~lynn/2003l.html#41> Secure OS Thoughts
<http://www.garlic.com/~lynn/2004c.html#7> IBM operating systems
<http://www.garlic.com/~lynn/2004g.html#4> Infiniband – practicalities for small clusters
<http://www.garlic.com/~lynn/2004.html#46> DE-skilling was Re: ServerPak Install via QuickLoad Product
<http://www.garlic.com/~lynn/2004.html#47> Mainframe not a good architecture for interactive workloads
<http://www.garlic.com/~lynn/2004l.html#16> Xah Lee's Unixism
<http://www.garlic.com/~lynn/2004n.html#4> RISCs too close to hardware?
<http://www.garlic.com/~lynn/2004n.html#25> Shipwrecks
<http://www.garlic.com/~lynn/2004n.html#34> RISCs too close to hardware?
<http://www.garlic.com/~lynn/2004o.html#20> RISCs too close to hardware?
<http://www.garlic.com/~lynn/93.html#23> MTS & LLMPS?
<http://www.garlic.com/~lynn/93.html#25> MTS & LLMPS?
<http://www.garlic.com/~lynn/93.html#26> MTS & LLMPS?
<http://www.garlic.com/~lynn/98.html#15> S/360 operating systems genealogy

--

Anne & Lynn Wheeler | <http://www.garlic.com/~lynn/>